Weakest Preconditions

Part 2: Calculating wp, wlp; Domain Predicates

9/29: p.4

A. Why

• Weakest liberal preconditions (wlp) and weakest preconditions (wp) are the most general requirements that a program must meet to be correct.

B. Objectives

At the end of today you should understand

• How to add error domain predicates to the wlp of a loop-free program to obtain its wp.

C. Calculating wlp for Loop-Free Programs

• It’s easy to calculate the wp and wlp of a loop-free/error-free program S especially since for such programs, the wp and wlp are identical.

• The following algorithm takes S and q and syntactically calculates a particular predicate for wlp(S, q), which is why it’s described using wlp(S, q) = ... instead of wp(S, q) ⇔ ....

  • wlp(skip, q) = q
  • wlp(v := e, Q(v)) = Q(e) where Q is a predicate function over one variable.
    • The operation that takes us from Q(v) to Q(e) is called syntactic substitution; we’ll look at it in more detail soon, but in the simple case, we simply inspect the definition of Q, search its text for occurrences of the variable v and replace them with copies of e.
  • wlp(S₁; S₂, q) = wlp(S₁, wlp(S₂, q))
  • wlp(if B then S₁ else S₂ fi, q) = (B → w₁) ∧ (¬B → w₂) where w₁ = wlp(S₁, q) and w₂ = wlp(S₂, q).
    • Since it’s equivalent, you can also use (B ∧ w₁) ∨ (¬B ∧ w₂).
  • wlp(if B₁ → S₁ ⊕ B₂ → S₂ fi, q) = (B₁ → w₁) ∧ (B₂ → w₂) where w₁ = wlp(S₁, q) and w₂ = wlp(S₂, q).
    • For the nondeterministic if, you must use (B₁ → w₁) ∨ (B₂ → w₂), not (B₁ ∧ w₁) ∨ (B₂ ∧ w₂), because they’re not equivalent (unlike the deterministic if statement).
    • When B₁ and B₂ are both true, either S₁ or S₂ can run, so we need B₁ ∧ B₂ → w₁ ∧ w₂, and this is implied by (B₁ → w₁) ∧ (B₂ → w₂).
    • Using (B₁ ∧ w₁) ∨ (B₂ ∧ w₂) fails because it allows for the possibility that B₁ and B₂ are both true but only one of w₁ and w₂ is true. This isn’t a problem when B₂ ⇔ ¬B₁, which is why we can use (B ∧ w₁) ∨ (¬B ∧ w₂) with deterministic if statements.
D. Some Examples of Calculating \( wp/wlp \):

- The programs in these examples never end in “state” \( \perp \), so the \( wp \) and \( wlp \) are equivalent.
- **Example 2:** \( wlp(x := x+1, x \geq 0) = x+1 \geq 0 \)
- **Example 3:** \( wlp(y := y+x; x := x+1, x \geq 0) \)
  \[ = wlp(y := y+x, wlp(x := x+1, x \geq 0)) \]
  \[ = wlp(y := y+x, x+1 \geq 0) = x+1 \geq 0 \]
- **Example 4:** \( wlp(y := y+x; x := x+1, x \geq y) \)
  \[ = wlp(y := y+x, wlp(x := x+1, x \geq y)) \]
  \[ = wlp(y := y+x, x+1 \geq y) \]
  \[ = x+1 \geq y+x \]
- If we were asked only to calculate the \( wlp \), we’d stop here. If we also wanted to logically simplify the \( wlp \) then \( x+1 \geq y+x \equiv y \leq 1 \).
- **Example 5:** (Swap the two assignments in Example 4)
  \[ wlp(x := x+1; y := y+x, x \geq y) \]
  \[ = wlp(x := x+1, wlp(y := y+x, x \geq y)) \]
  \[ = wlp(x := x+1, x \geq y+x)) \]
  \[ = x+1 \geq y+x+1 \text{ [if you want to logically simplify]} \]
- **Example 6:** \( wlp if y \geq 0 \text{ then } x := y fi, x \geq 0 \)
  \[ = wlp if y \geq 0 \text{ then } x := y else skip fi, x \geq 0 \]
  \[ = (y \geq 0 \implies wlp(x := y, x \geq 0)) \land (y < 0 \implies wlp(skip, x \geq 0)) \]
  \[ = (y \geq 0 \implies y \geq 0) \land (y < 0 \implies x \geq 0) \]
- It’s also okay to use \( (y \geq 0 \land y \geq 0) \lor (y < 0 \land x \geq 0) \).
- If we want to simplify logically, we can continue with
  \[ \equiv y \geq 0 \lor (y < 0 \land x \geq 0) \]
  \[ \equiv (y \geq 0 \lor y < 0) \land (y \geq 0 \lor x \geq 0) \]
  \[ \equiv y \geq 0 \lor x \geq 0 \text{ (which is also } \equiv y < 0 \implies x \geq 0 \text{, if you prefer)} \]

E. Avoiding Runtime Errors in Expressions

- To avoid runtime failure of \( \sigma(e) \), we’ll take the context in which we’re evaluating \( e \) and augment it with a predicate that guarantee non-failure of \( \sigma(e) \). For example, for \( \{ P(e) \} v \mathrel{:=} e \{ P(v) \} \), we’ll augment the precondition to guarantee that evaluation of \( e \) won’t fail.
- For each expression \( e \), we will define a **domain predicate** \( D(e) \) such that \( \sigma \models D(e) \) implies \( \sigma(e) \neq \perp_e \).
  - This predicate has to be defined recursively, since we need to handle complex expressions like \( D(b[b[k]]) = 0 \leq k < \text{size}(b) \land 0 \leq b[k] < \text{size}(b) \).
  - As with \( wp \) and \( sp \), the domain predicate for an expression is unique only up to logical equivalence. For example, \( D(x/y + u/v) = y \neq 0 \land v \neq 0 \iff v \neq 0 \land y \neq 0 \).
• **Definition** (Domain predicate \(D(e)\) for expression \(e\)) We must define \(D\) for each kind of expression that can cause a runtime error:

  - \(D(c) = D(v) = T\) if where \(c\) is a constant and \(v\) is a variable.
  - Evaluation of a variable or constant doesn't cause failure.
  - \(D(b[e]) = D(e) \land 0 \leq e < \text{size}(b)\)
  - \(D(e_1/e_2) = D(e_1 \% e_2) \iff D(e_1) \land D(e_2) \land e_2 \neq 0\)
  - \(D(\sqrt(e)) = D(e) \land e \geq 0\)
  - And so on, depending on the datatypes and operations being used.

  - The various operations (+, -, etc.) and relations (\(\leq\), =, etc.) don't cause errors but we still have to check their subexpressions:

  - \(D(e_1 \text{ op } e_2) = D(e_1) \land D(e_2)\), except for \text{ op } = / or \%
  - \(D(\text{ op } e) = D(e)\), unless you add an operator that can cause runtime failure.

  - \(D(\text{if } B \text{ then } e_1 \text{ else } e_2 \text{ fi}) = D(B) \land (B \rightarrow D(e_1)) \land (\neg B \rightarrow D(e_2))\)

    • (For a conditional expression, we only need safety of the one branch we execute.)

  - **Example 7**: \(D(b[b[k]]) = D(b[k]) \land 0 \leq b[k] < \text{size}(b)\)

    \[\equiv 0 \leq k < \text{size}(b) \land 0 \leq b[k] < \text{size}(b)\]

  - **Example 8**: \(D((b + \sqrt(b^*b - 4*a^*c))/(2*a))\)

    \[= D(e) \land D(2*a) \land 2*a \neq 0\]

    where \(e = -b + \sqrt(b^*b - 4*a^*c)\)

    \[= D(-b) \land D(\sqrt(b^*b - 4*a^*c)) \land D(2*a) \land 2*a \neq 0\]

    \[\iff D(\sqrt(b^*b - 4*a^*c)) \land 2*a \neq 0\]

    since \(D(-b) = D(2*a) = T\)

    \[= D(b^*b - 4*a^*c) \land (b^*b - 4*a^*c \geq 0) \land 2*a \neq 0\]

    \[\iff b^*b - 4*a^*c \geq 0 \land 2*a \neq 0\]

  - **Example 9**: \(D(\text{if } 0 \leq k < \text{size}(b) \text{ then } b[k] \text{ else } 0 \text{ fi})\)

    \[= D(B) \land (B \rightarrow D(b[k]) \land (\neg B \rightarrow D(0))\]

    where \(B = 0 \leq k < \text{size}(b)\)

    \[= (B \rightarrow D(b[k])) \land (\neg B \rightarrow T)\]

    since \(D(B)\) and \(D(0) = T\)

    \[\iff B \rightarrow D(b[k])\]

    since everything implies true

    \[= B \rightarrow D(k) \land 0 \leq k < \text{size}(b)\]

    expanding \(D(b[k])\)

    \[\iff B \rightarrow T \land B\]

    since \(B = 0 \leq k < \text{size}(b)\)

    \[\iff T\]

---

**F. Avoiding Runtime Errors in Programs**

• Recall that we extended our notion of operational semantics to include \(\langle S, \sigma \rangle \rightarrow^* \langle E, \bot_e \rangle\) to indicate that evaluation of \(S\) causes a runtime failure.

• We can avoid runtime failure of statements by adding domain predicates to the preconditions of statements. Though for loops we can't in general calculate the \(wlp/wp\), we can calculate the domain predicate for them.
• **Definition:** For statement $S$, the predicate $D(S)$ gives a sufficient condition to avoid runtime errors.
  
  - $D(\text{skip}) = T$
  - $D(v := e) = D(e)$
  - $D(b[e] := e_1) = D(b[e_1]) \land D(e_2)$
  - $D(S_1 ; S_2) = D(S_1) \land wp(S_1, D(S_2))$
    - $\sigma \models D(e)$ iff $\sigma(e) \neq \perp_e$
    - $\sigma \models D(S)$ iff $\langle S, \sigma \rangle \not\models \langle E, \perp_e \rangle$
    - If $\sigma \models D(S_1)$ then $\perp_e \not\in M(S_1, \sigma)$
    - If $\sigma \models wp(D(S_2))$, then $M(S_1, \sigma) \models D(S_2)$
      - Either way, $\perp \not\in M(S_1, M(S_1, \sigma))$
  - $D(\text{if } B \text{ then } S_1 \text{ else } S_2 \text{ fi}, q) = D(B) \land (B \rightarrow D(S_1)) \land (\neg B \rightarrow D(S_2))$
  - $D(\text{if } B_1 \rightarrow S_1 \land B_2 \rightarrow S_2 \text{ fi}, q) = D(B_1 \lor B_2) \land (B_1 \lor B_2) \land (B_1 \rightarrow D(S_1)) \land (B_2 \rightarrow D(S_2))$
    - The condition $(B_1 \lor B_2)$ avoids failure of the nondeterministic if-fi due to none of the guards holding. This definition extends easily to if-fi with more than two guarded commands.
  - $D(\text{while } B \text{ do } S_1 \text{ od}) = D(B) \land (B \rightarrow D(S_1))$
  - $D(\text{do } B_1 \rightarrow S_1 \land B_2 \rightarrow S_2 \text{ od}) = D(B_1 \lor B_2) \land (B_1 \lor B_2) \land (B_1 \rightarrow D(S_1)) \land (B_2 \rightarrow D(S_2))$
    - The domain predicate for nondeterministic do-od is like that for if-fi except that having none of the guards holds does not cause an error.
    - With the domain predicates, it’s easy to extend $\text{wp}$ for $wp$ for loop-free programs because we don’t have to argue for termination of a loop.

  • [9/29 mild rephrasing] **Definition:** $wp(S, q) = D(S) \land D(w) \land w$, where $w = wp(S, q)$.

  • **Example 10:** If a program does a division, then the $wp$ and $wlp$ can differ.
    - Let $p_2 = wp(x := y; z := v/x, z > x + 2) = wp(x := y, p_1)$
    - Where $p_1 = wp(z := v/x, z > x + 2) = D(z := v/x) \land D(w) \land w$
      - where $w = wp(x := v/x, z > x + 2) = v/x > x + 2$
      - $p_1 = D(z := v/x) \land D(v/x > x + 2) \land v/x > x + 2 = x \neq 0 \land x \neq 0 \land v/x > x + 2 \iff x \neq 0 \land v/x > x + 2$
    - So $p_2 = wp(x := y, p_1) = wp(x := y, x \neq 0 \land v/x > x + 2)$
      - $= wp(x := y, x \neq 0 \land v/y > y + 2)$, since $x := y$ causes no errors
      - $= y \neq 0 \land v/y > y + 2$

  • **Example 11:** Let’s calculate $p_0 = wp(x := b[k], \sqrt{x} \geq 1)$. When $S = x := b[k]$ and $q = \sqrt{x} \geq 1$, then
    - $p_0 = wp(S, q) = D(S) \land D(w) \land w$
      - where $w = wp(S, q) = wp(x := b[k], q = \sqrt{x} \geq 1) = \sqrt{b(k)} \geq 1 \quad [\text{end 9/28 rephrasing}]
    - Breaking this down,
      - $wp(S, q) = wp(x := b[k], \sqrt{x} \geq 1) \iff \sqrt{b(k)} \geq 1$, so
      - $D(wp(S, q)) = D(\sqrt{b(k)} \geq 1) = D(b[k]) \land b[k] \geq 0 \land k < \text{size}(b) \land b[k] \geq 0$. 

© James Sasaki, 2020
• \( D(S) = D(x := b[k]) = D(k) \land 0 \leq k < \text{size}(b) \iff 0 \leq k < \text{size}(b) \)

• Combining, we get

\[
wp(x := b[k], \sqrt{x} \geq 1) \\
= D(wp(x := b[k], \sqrt{x} \geq 1)) \land wp(x := b[k], \sqrt{x} \geq 1) \land D(x := b[k]) \\
\iff (\sqrt{b[k]} \geq 1) \land (0 \leq k < \text{size}(b) \land b[k] \geq 0) \land (0 \leq k < \text{size}(b)) \\
\iff 0 \leq k < \text{size}(b) \land b[k] \geq 0 \land \sqrt{b[k]} \geq 1 \\
\iff 0 \leq k < \text{size}(b) \land b[k] \geq 1 
\text{ (if we decide to simplify numerically)}